**北京邮电大学课程设计报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | **面向对象程序设计实践（C++）** | **学 院** | **计算机学院** | **指导教师** | **邹华** |
| **学生姓名** | **刘立敏** | **班 级** | **2018211310** | **学 号** | **2018211398** |
| **课**  **程**  **设**  **计**  **内**  **容** | 简要介绍课程设计的主要内容，包括课程设计教学目的、基本内容、实验方法和团队分工等  设计内容：使用面向对象的设计方法来设计一款平台类对战游戏——宠物小精灵对战系统。具体包括宠物小精灵的类的设计，使用socket进行客户端与服务器的通信，从而实现用户注册与平台登陆，使用数据库对用户和小精灵的各种信息进行存储，以及最后实现游戏对战的设计。  实验方法：基于Qt开发图形界面，基于UDP协议进行客户端与服务器的通信，使用SQLite数据库  团队分工：单人完成。 | | | | |
| **学生**  **课程设计**  **报告**  （附页） | 见程序代码及课程设计报告。 | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | 遵照实践教学大纲并根据以下四方面综合评定成绩：  1、课程设计目的任务明确，选题符合教学要求，份量及难易程度  2、团队分工是否恰当与合理  3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果  4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范  **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | |

目录

[设计任务描述 3](#_Toc50372100)

[阶段一 5](#_Toc50372101)

[1、总体设计 5](#_Toc50372102)

[2、详细设计 6](#_Toc50372103)

[2.1、宝可梦类 6](#_Toc50372104)

[2.2、交互界面 7](#_Toc50372105)

[3、实现效果 7](#_Toc50372106)

[阶段二 9](#_Toc50372107)

[1、总体设计 9](#_Toc50372108)

[2、详细设计 9](#_Toc50372109)

[2.1、C/S模式 9](#_Toc50372110)

[2.2、数据库 12](#_Toc50372111)

[3、实现效果 14](#_Toc50372112)

[3.1、登录界面 14](#_Toc50372113)

[3.2、主界面 15](#_Toc50372114)

[3.3、玩家信息界面 16](#_Toc50372115)

[3.4、宝可梦属性界面 17](#_Toc50372116)

[阶段三 18](#_Toc50372117)

[1、总体设计 18](#_Toc50372118)

[2、详细设计 19](#_Toc50372119)

[2.1、战斗模式选择与参战宝可梦选择 19](#_Toc50372120)

[2.2、战斗规则设计 19](#_Toc50372121)

[2.3、伤害计算方式与技能 20](#_Toc50372122)

[2.4、成就系统 22](#_Toc50372123)

[2.5、动画效果 22](#_Toc50372124)

[3、实现效果 23](#_Toc50372125)

[3.1、前文已介绍的界面 23](#_Toc50372126)

[3.2、成就界面 24](#_Toc50372127)

[3.3、战斗选择界面 25](#_Toc50372128)

[3.4、战斗界面 26](#_Toc50372129)

[3.5、战斗结算界面 27](#_Toc50372130)

[心得体会 29](#_Toc50372131)

[附录1：战斗规则 30](#_Toc50372132)

[附录2：开发日志 33](#_Toc50372133)

# 设计任务描述

**题目一：宠物小精灵的加入**

要求：

 设计宠物小精灵的类，为简化游戏设计，精灵的属性包括种类（力量型：高攻击；肉盾型：高生命值；防御型：高防御；敏捷型：低攻击间隔，共四种）、名字、等级、经验值、攻击力、防御力、生命值、攻击间隔等（以上属性必须，其他属性可自行添加）（基本要求：本游戏中只有上面的4种类型。进一步要求：上述4种类型可以进一步深入划分，比如力量型又可以再细分为：沙瓦朗、火爆猴、腕力等）

 每个精灵初始等级为1，满级15级，每当精灵升级的时候，宠物对应的属性值会有少量增加（主属性增加量相对较多）

 每个精灵有自己独特的攻击方式，如“闪电攻击”，“火焰攻击”等等，请设计一个精灵的基类，并将精灵的攻击方法设为虚方法以方便子类重写

 请写一个测试程序对设计的精灵类的相关属性和方法（包括攻击函数，升级函数等）进行测试

 题目主要考察点：类的继承，对象数据成员设计，成员函数设计

**题目二：用户注册与平台登录**

要求：

 每个用户需要注册一个账号，用户名全局唯一，不能有任何两个用户名相同，要考虑注册失败的场景时的反馈

 实现注册、登录、登出功能，均采用C/S模式，客户端和服务端用socket进行通信，服务端保存所有用户的信息

 每个用户拥有：用户名、拥有的精灵，两个属性。用户注册成功时，系统自动随机分发三个1级精灵给用户

 用户可以查看所有成功注册用户拥有的精灵，也可以查看所有当前在线的用户

 题目主要考察点：socket通信，交互场景反馈，用户信息存储方式，界面交互，其它合理的新颖设计。

**题目三：游戏对战的设计**

要求：

 已经登录的在线用户可以和服务器进行虚拟决斗，决斗分两种：升级赛和决斗赛，两种比赛都能增长精灵宠物经验值。服务器上有一个虚拟精灵的列表，用户可以挑选其中任意一个进行比赛（升级赛或者决斗赛）。另外决斗赛中用户胜出可以直接获得该战胜的精灵，失败则系统从用户的精灵中随机选三

个（不够三个精灵的情况就选择他所有的精灵），然后由用户选一个送出。

 升级赛只是用户用来增加精灵经验值，规则开发者自定；

 累积多少经验值升一级，规则开发者自定；

 决斗赛的上述规则同升级赛，只是额外还可以赢得一个宠物或失去一个宠物。

 用户如果没有精灵（比如总是失败，已经全部送出去），则系统会随机放给他一个初级精灵。

 请让你的系统自动模拟每场比赛的每次出招。另外，为了增加不确定性，可以加入概率闪避攻击和暴击伤害机制

 比赛的过程和结果由系统根据上述规则自动模拟完成，要求结果具有一定的随机性。

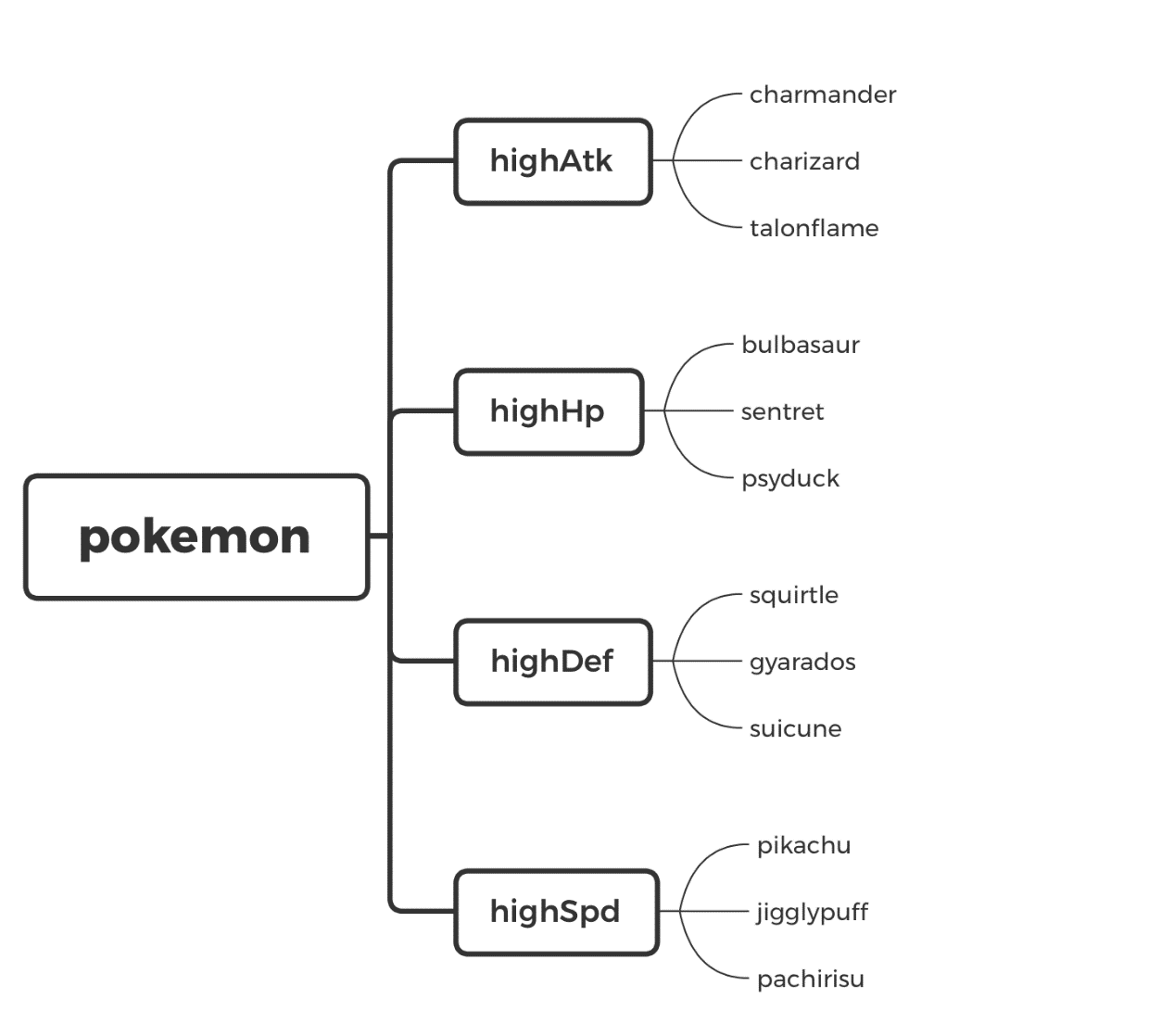
 用户增加新功能，可以查看某个用户的胜率

 用户增加新属性，为宠物个数徽章（金银铜）和高级宠物徽章（金银铜），分别根据拥有的宠物个数的多少和拥有高级宠物（15级）个数的多少颁发

 题目主要考察点：客户端与服务器数据交互（可采用多进程或异步通信或其他方法均可），并发请求处理，类的方法设计，伤害计算方法设计，界面交互，其它合理的新颖设计。

# 阶段一

## 1、总体设计

****

①上图中，pokemon是基类名，中间的highAtk(力量型：高攻击)，highHp（肉盾型：高生命）,highDef（防御型：高防御）,highSpd（敏捷型：高攻速）为pokemon的四个派生类。右边的12个类又是他们的派生类，表示宝可梦的种类名。

②基类中的属性包括了种类、名字、等级、经验值、攻击力、防御力、最大生命值、当前生命值、攻速（攻击间隔）、技能等。

③基类中的方法包括设置初始属性、获取各种属性、攻击、承受伤害、获取经验升级、判断是否死亡等等。其中pokemon类和他的四个派生类的攻击方法被设置为虚方法，由更下一层的12个子类重写来具体实现。在第一阶段，我们将攻击函数仅仅设计为输出攻击技能名称，真正实现攻击的任务将在第三阶段的设计中实现。

## 2、详细设计

### 2.1、宝可梦类

class **pokemon** : public QObject //宝可梦类

{

Q\_OBJECT

public:

**pokemon**(); //构造函数

~***pokemon***(); //析构函数

virtual QString ***attack***()=0; //攻击函数

int **gainExp**(uint Exp); //获得经验函数

int **takeDamage**(uint dmg); //承受伤害函数

bool **judgeDeath**(); //判断死亡

uint **getLevel**(){return level;} //获取等级信息

uint **getExp**(){return exp;} //获取经验值信息

uint **getAtk**(){return atk;} //获取攻击值

uint **getMaxHp**(){return maxHp;} //获取最大生命值

uint **getHp**(){return hp;} //获取生命值信息

uint **getDef**(){return def;} //获取防御值

uint **getSpd**(){return spd;} //获取攻速值

void **setAttributeValue**(QString Type,QString Name,uint Atk,uint Def,uint MaxHp,int Hp,uint Spd); //设置宝可梦属性值

protected:

QString type; //种类

QString name; //名字

uint level; //等级

uint exp; //经验值

uint atk; //攻击值

uint def; //防御值

uint maxHp; //最大生命值

int hp; //当前生命值

uint spd; //攻速

QString skill; //技能

};

### 2.2、交互界面

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = 0);

~***MainWindow***();

void **getPokemon**(); //获取宝可梦

void **updateHp**(uint hp,uint maxHp); //更新生命值

void **updateLevelAndExp**(uint level, uint exp); //更新等级经验值

void **updateAttributeValue**(uint atk,uint maxHp,uint def,uint spd); //更新属性值

private slots:

void **on\_attackBtn\_clicked**(); //攻击测试

void **on\_takeDamageBtn\_clicked**(); //承受伤害测试

void **on\_gainExpBtn\_clicked**(); //获得经验测试

void **on\_clear\_clicked**(); //清空状态栏

void **on\_getPokemonBtn\_clicked**(); //获取/更换宝可梦测试

void **on\_resetEnemy\_clicked**(); //重置敌人生命值

private:

Ui::MainWindow \*ui;

pokemon\* myPokemon;

};

## 3、实现效果

如下图所示，界面分为三个部分，依次为上方的测试函数（包括获取/更换宝可梦函数，攻击函数，承受攻击函数和获得经验函数），左边的各属性展示，以及右边的状态信息展示栏（包括攻击、受击、升级等操作）。



①点击“攻击”、“承受攻击”、“获得经验”、“获取/更换宝可梦”按钮，依次执行对应的操作，下方的属性栏和信息状态栏随即更新信息。

攻击：我方宝可梦发动技能降低敌方生命值。

承受攻击：敌方宝可梦发动攻击降低我方生命值。

获得经验：增加经验值。

获取/更换宝可梦：获得一只初始状态的宝可梦。

②点击“重置”按钮，敌方生命值恢复满（默认300点）。

③点击“clear”按钮，信息状态栏清空。

④当敌方生命值降为0时，敌方宝可梦死亡。此时无法执行“攻击”和“承受攻击”操作，只有“重置”敌方生命值后才可进行。

⑤当我方生命值降为0时，我方宝可梦死亡。此时无法执行“攻击”、“承受攻击”和“获得经验”操作。点击“获取/更换宝可梦”按钮可以恢复。

⑥获取经验值后，如果经验槽满了将会升级，升级后各项属性得到提升（主属性会有额外提升），同时生命值恢复满。

# 阶段二

## 1、总体设计

本阶段采用了C/S模式设计，客户端展现用户与系统的交互界面，服务器建立了数据库存储用户数据，并响应客户端的请求。

用户需要注册一个用户名全局唯一的账号，然后登录进入主界面。在主界面里，用户可以执行查看自己的宝可梦、查看所有的在线用户以及他们的宝可梦、登出等操作。

## 2、详细设计

### 2.1、C/S模式

本阶段开始采用C/S模式进行设计。使用UDP协议，客户端和服务器之间通过QUdpSocket进行通信。

#### 2.1.1、建立通信

客户端：

udpSocket=new QUdpSocket(this);

QHostAddress sender=udpSocket->localAddress();

quint16 senderPort=udpSocket->localPort();

udpSocket->bind(sender,senderPort); connect(udpSocket,SIGNAL(readyRead()),this,SLOT(readPendingDatagrams()));

udpSocket绑定本地地址localAddress和本地端口localPort。当服务器有数据报传过来时，产生readyRead信号，从而触发readPendingDatagrams槽函数，进而对数据报进行处理。

服务器：

udpSocket=new QUdpSocket(this); //与客户端之间使用UDP通信

udpSocket->bind(QHostAddress::LocalHost,8888); //本地地址，端口8888 connect(udpSocket,SIGNAL(readyRead()),this,SLOT(readPendingDatagrams())); //建立连接

udpSocket绑定localHost和端口8888。当客户端有请求时，产生readyRead信号，从而触发readPendingDatagrams槽函数，进而对数据报进行处理。

#### 2.1.2、数据报格式

操作类型：

enum

**state**{SIGN\_IN,SIGN\_UP,GET\_POKEMON\_INFO,GET\_ALL\_PLAYERS,GET\_OTHER\_POKEMON\_INFO,UPDATE\_DATABASE};

//操作类型：登录/注册/获取宝可梦信息/查看所有玩家信息/获取其他玩家的宝可梦信息/更新数据库

客户端：

struct **user**

{

state clientState; //操作类型

QString userName; //用户名

QString password; //密码

QVector<pokemon\*>pokemonInfo; //宝可梦仓库，用于更新数据库

};

clientState向服务器指明操作类型；

userName和password用于向服务器请求注册或登录，userName还可用于查询其他玩家的宝可梦信息。

pokemonInfo提供当前客户端宝可梦的本地数据给服务器，用于同步更新服务器端的数据库。

服务器：

struct **message**

{

state serverState; //操作类型

QVector<pokemon\*>pokemonInfo; //宝可梦仓库

QVector<user\*>userInfo; //用户列表

};

serverState向客户端指明操作类型；

pokemonInfo保存了从服务器数据库取出的宝可梦信息，提供给客户端；

userInfo保存了从服务器数据库取出的在线用户信息，提供给客户端；

#### 2.1.3、发送、接收数据报

**（以更新服务器数据库为例）**

用户在登出账号或退出游戏时，系统会自动将本地数据上传至服务器。客户端负责将当前的宝可梦信息封装进数据报并发送，服务器则接受数据报并以此更新数据库。

客户端：

user \*oneUser=new user;

oneUser->userName=currentUserName; //客户端当前的用户名

oneUser->clientState=UPDATE\_DATABASE; //向服务器指明当前操作为请求更新数据库

oneUser->pokemonInfo=pokemonVector; //更新后的宝可梦数据

//封装数据报

QByteArray byteArray;

byteArray.append((char\*)oneUser,sizeof(user));

udpSocket->writeDatagram(byteArray,QHostAddress::LocalHost,8888); //将数据报发送给服务器

服务器：

QHostAddress sender; //IP地址

quint16 senderPort; //端口号

while(udpSocket->hasPendingDatagrams()) //收到数据报

{

user oneUser; udpSocket->readDatagram((char\*)&oneUser,sizeof(oneUser),&sender,&senderPort); //读取数据报信息到oneUser中

……

if(oneUser.clientState==UPDATE\_DATABASE) //更新数据库

{

QVector<pokemon\*>pokemonVector=oneUser.pokemonInfo;

QSqlQuery query;

……(更新数据库操作)

}

}

### 2.2、数据库

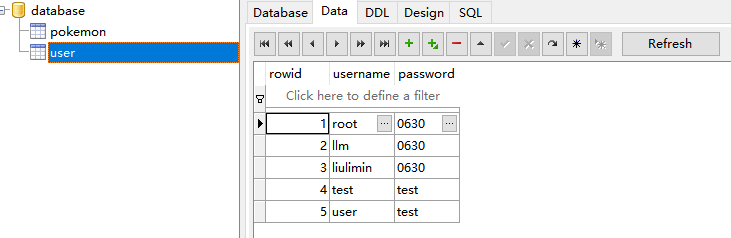
本系统使用了SQLITE数据库。

服务器端建立了两个数据库，分别为用户数据库（user）和宝可梦数据库（pokemon）。

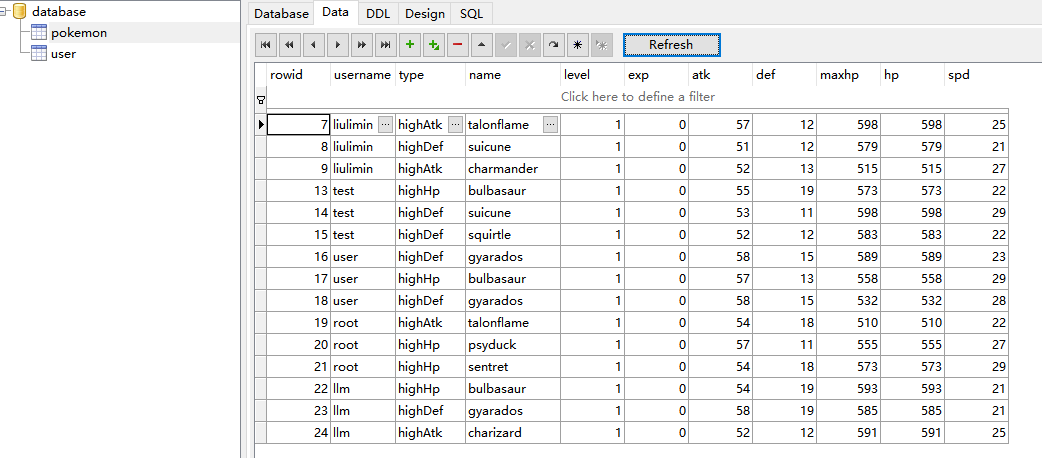
用户数据库用来存储用户的账号和密码信息；宝可梦数据库存储了宝可梦所属的用户和各项属性信息。两个数据库都采用用户名（userName）作为主关键字。

用户登录或注册时，服务器从用户数据库中获取账号信息，确认是否注册或登录成功。然后从宝可梦数据库中获取该用户拥有的宝可梦信息，发送给客户端。当用户在客户端请求查看他人的宝可梦数据时，服务器也会查询数据库并返回所需信息。当用户执行登出操作或退出游戏时，服务器端的宝可梦数据库将自动同步客户端的数据。

**数据库样例展示**



上图为用户账号数据库，表格数据有username和password两项。



上图为宝可梦数据库，表格数据有

username（用户名），

type（类型），

name（名字），

level（等级），

exp（经验值），

atk（攻击力），

def（防御力），

maxhp（最大生命值），

hp（生命值），

spd（攻击速度）

共十项属性

## 3、实现效果

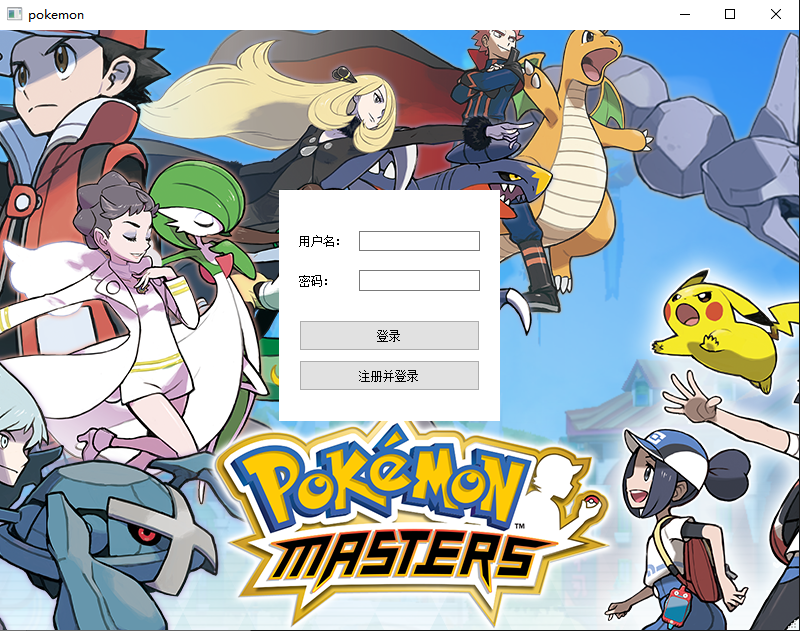
### 3.1、登录界面

①用户输入用户名和密码，执行注册或登陆操作（用户名和密码均不能为空）

②用户名是全局唯一的，当输入的用户名已经在数据库里时，注册将会失败，系统提示用户更换用户名进行注册或直接登录该账号。

③密码错误时，系统弹出错误框进行提示。

④成功注册或正确输入密码后，进入主界面。



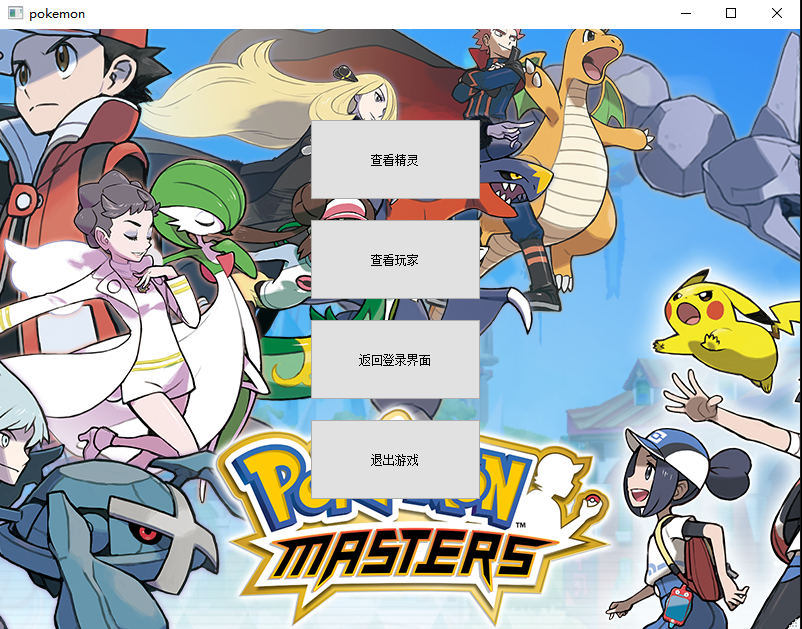
### 3.2、主界面

①查看精灵：查看自己拥有的宝可梦数据，进入宝可梦属性界面。

②查看玩家：查看所有注册成功过的用户，进入玩家信息界面。

③返回登录界面：执行登出操作，回到登录界面

④退出游戏：关闭界面。



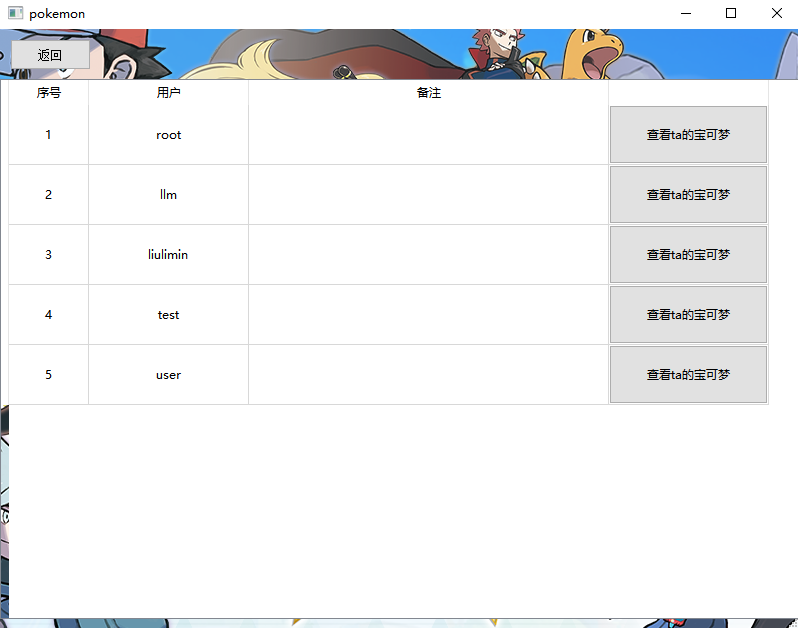
### 3.3、玩家信息界面

①表格共4列，分别为序号、用户名、备注以及“查看ta的宝可梦”按钮。

②备注一列暂时留空，方便阶段三添加新功能。

③返回：返回主界面。

④查看ta的宝可梦：查看该用户的宝可梦信息，进入宝可梦属性界面

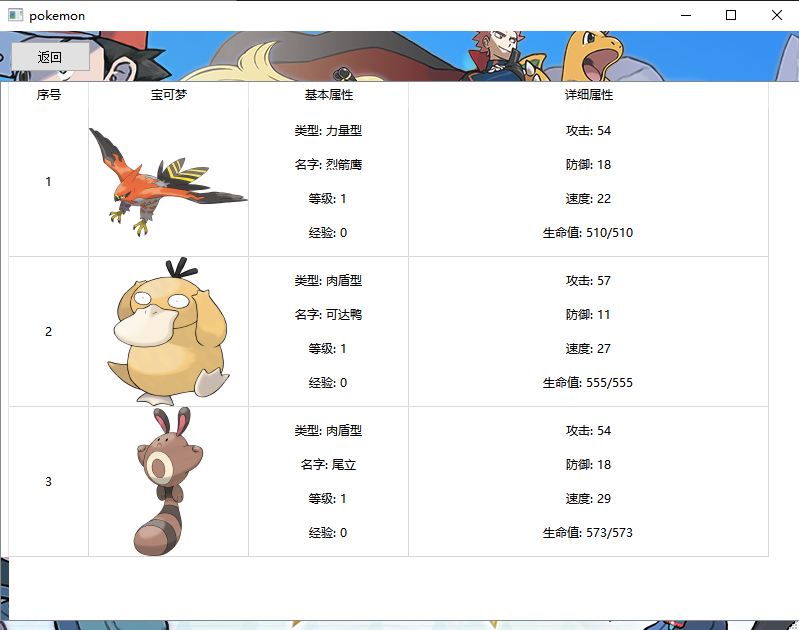


### 3.4、宝可梦属性界面

①查看自己的宝可梦和查看别人的宝可梦都会进入该界面。

②点击返回：返回上一界面（可能是主界面，也可能是玩家信息界面）

③表格共4列，依次为序号，宝可梦图片，基本属性（包含类型、名字、等级、经验）和详细属性（包含攻击、防御、速度、生命）。



# 阶段三

## 1、总体设计

本阶段的任务主要为游戏对战的设计，包括双方参战宝可梦的选择、战斗规则设计、伤害计算方式设计、技能设计以及相关的界面交互设计。

此外，还有在之前阶段基础上，新增了查看用户历史的交战记录（包括胜负场次、胜率）的功能，以及徽章展示功能。我将这两者合并为成就系统。

## 2、详细设计

### 2.1、战斗模式选择与参战宝可梦选择

用户进入战斗后，依次选择自己所拥有的宝可梦和服务器的一只宝可梦进行虚拟决斗。

战斗模式分两种：升级赛和决斗赛。升级赛仅仅通过战斗使得宝可梦获得经验，从而提升等级和各项属性值；决斗赛除了经验值外，用户如果战斗胜利，可以获得该战胜的宝可梦。而一旦失败，就要选择一只宝可梦送出。

在参战宝可梦选择界面里，将会给出一张表，上面列出了用户所拥有的全部宝可梦的所有信息，用户需要选择一只参战。然后服务器会随机生成15只宝可梦，等级分别为1到15级各不相同。用户可以根据自己当前宝可梦的能力值来选择作为敌方的宝可梦进行对战。

### 2.2、战斗规则设计

本游戏采用回合制战斗方式。

战斗中每回合双方互相攻击一次。双方的速度属性决定攻击的先后手顺序。

（来自开发者的题外话：原本设计构想里速度是一项重要的属性，决定双方宝可梦的攻击频率。但是在实际开发的过程中，发现这项属性对技能释放，尤其是战斗的动画演示有相当大的影响，给游戏开发的复杂程度带来了几何级的增长。经过反复尝试失败后的综合考虑，我决定放弃对速度属性的仔细探究。于是，速度属性的重要性被大大弱化，目前仅仅用来决定攻击先后手顺序。然而，这终究是个遗憾，今后有机会的话我一定会想办法弥补）

战斗中的攻击有两种——普通攻击和释放技能。普通攻击只要在我方回合均可以使用，技能释放则有限制：战斗引入了怒气值系统，怒气值在0-5之间；初始怒气值为0，受到攻击后可以随机获得1-2点怒气值；当怒气值积满（5点）时，则可以释放一次技能，同时清空怒气值。

战斗双方只要有任何一方生命值归0时，战斗结束。

在游戏中，用户可以选择进行自动战斗。开启自动战斗后，系统默认提高技能释放的优先级（即：怒气值满后立即释放技能）。

更加详细的战斗规则请参见 [附录：战斗规则](#_附录1：战斗规则)。（按住ctrl单击）

### 2.3、伤害计算方式与技能

具体内容请参见 [附录：战斗规则](#_附录1：战斗规则)。

这里以尾立（sentret，宝可梦名）的技能为例介绍。

int sentret::***useSkill***(skillName &name) //顽强：每损失3%最大生命，下次攻击造成的伤害提高10%（土系）

{

int hp1=this->getMaxHp(); //最大生命

int hp2=hp1-this->getHp(); //当前已损失生命

int percentage=hp2\*100/hp1; //损失生命所占百分比

double skillRatio=int(percentage/3)\*0.1+1; //技能倍率

int atk=this->getAtk();

double ratio=1+double(QRandomGenerator::global()->bounded(-10,11))/100;

//攻击比率为0.9~1.1

int realAtk=atk\*ratio\*skillRatio; //面板攻击\*基础倍率\*技能倍率

double criticalProbability=0.2; //暴击概率0.2

double criticalRatio=1.5; //暴击伤害倍率1.5

double n=QRandomGenerator::global()->bounded(0,10); //n是0~9之间的随机数

if(n<int(criticalProbability\*10)) //n为0或1时为真，对应20%概率触发暴击

{

realAtk\*=criticalRatio;

qDebug()<<"暴击！";

}

name=TENACITY;

return realAtk; //最终攻击

}

函数int sentret::***useSkill***(skillName &name)中

skillName是一个枚举类型；函数参数name是一个枚举变量的引用，表示该技能的名称，用于特殊技能效果的处理和UI界面展示；函数返回这次技能最终造成的攻击伤害，是int类型。

如技能描述——顽强：每损失3%最大生命，下次攻击造成的伤害提高10%。

在该函数中，首先计算损失的生命值，以此来计算技能的伤害倍率。

然后计算技能的伤害=面板攻击\*基础倍率\*技能倍率。

最后，再考虑暴击机制，通过随机数生成来判断是否触发暴击，如果触发，则最终技能伤害再提升至1.5倍。

下面是承受伤害函数。

int pokemon::**takeDamage**(int dmg,bool &dodge) //承受伤害函数

{

int n=QRandomGenerator::global()->bounded(0,10); //n是0~9之间的随机数

double dodgeProbability=0.1; //闪避率10%

if(n<int(dodgeProbability\*10)) //n为0时闪避

{

qDebug()<<"闪避";

dodge=true;

return 0;

}

else dodge=false;

int tmp=dmg-def; //敌方攻击值减去我方防御值

if(tmp>0) //敌人攻击值高于我方防御值，造成伤害

{

if(hp<tmp) //我方死亡

tmp=hp; //造成的伤害只能为死亡前所剩生命值

hp-=tmp;

}

else //敌方攻击未能破防

tmp=0;

return tmp;

}

在之前的使用技能（useSkill）函数中，返回值传到takeDamage函数的第一个参数dmg里。dmg是技能（或者普攻）的最终攻击值，减去承受攻击方的防御值后，才是受击方最终所受伤害。此外，还需进行闪避判定，如果判定成功，则不受任何伤害。

dodge是bool型变量的引用，用来标记是否触发闪避。

### 2.4、成就系统

系统将记录用户历史对战的胜负场次和胜率。用户可以在成就系统界面查看。在查看其它用户的信息时，也可以看到对方的相关记录。

成就界面里还可以查看到自己拥有的徽章。徽章分为两种：宝可梦徽章和高级宝可梦徽章。每种徽章又分为铜、银、金三个等级。宝可梦徽章在拥有5、10、15只宝可梦时取得，高级宝可梦徽章在拥有5、10、15只满级（15级）宝可梦时取得。

### 2.5、动画效果

#### 2.5.1、子弹移动

void Client::**updateMyBullet**() //更新我方子弹坐标

{

QPoint point=ui->ui6\_bullet->pos(); //获取当前子弹坐标

if(point.x()<=720)

{

ui->ui6\_bullet->setGeometry(point.x()+10,point.y(),50,50); //子弹右移，重置坐标

}

}

更新子弹坐标函数如上所示，其中ui->ui6\_bullet是QLabel标签控件，我们给该控件添加上子弹的图片，所以通过pos()函数可以获取当前子弹的坐标位置。

point.x()和point.y()分别表示point的横纵坐标，setGeometry(int x,int y,int w,int h)函数用于设置控件的坐标（x,y,w,h依次表示横坐标，纵坐标，宽度和高度）。

因此，该函数的作用就是使得子弹坐标右移十个单位。

timer=new QTimer;

connect(timer,SIGNAL(timeout()),this,SLOT(updateMyBullet()));

timer->setTimerType(Qt::PreciseTimer);

timer是一个定时器，我们将其的timeout()信号绑定到updateMyBullet()槽函数上，然后设置timer->start(30),表示每30ms产生一次timeout()信号，从而触发执行updateMyBullet函数，更新子弹的坐标。

综上所述，就可以实现每30ms一次使得子弹坐标右移10个单位，从而展现出子弹攻击的效果。

#### 2.5.2、伤害、暴击、闪避、技能、怒气值显示

每当子弹移动到受击方面前时，就被设置为不可见，同时在受击方的上方显示本次受到的伤害数字。

释放技能时，界面上会显示技能名称，以示与普攻的区别。

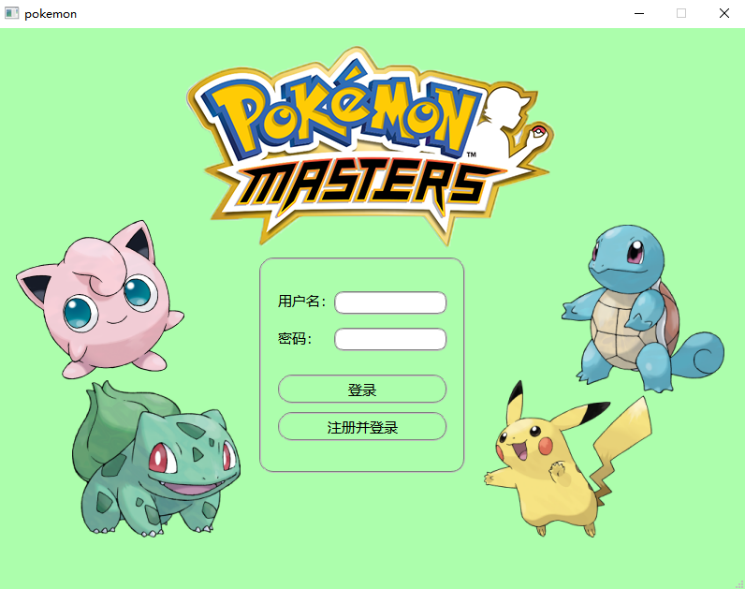
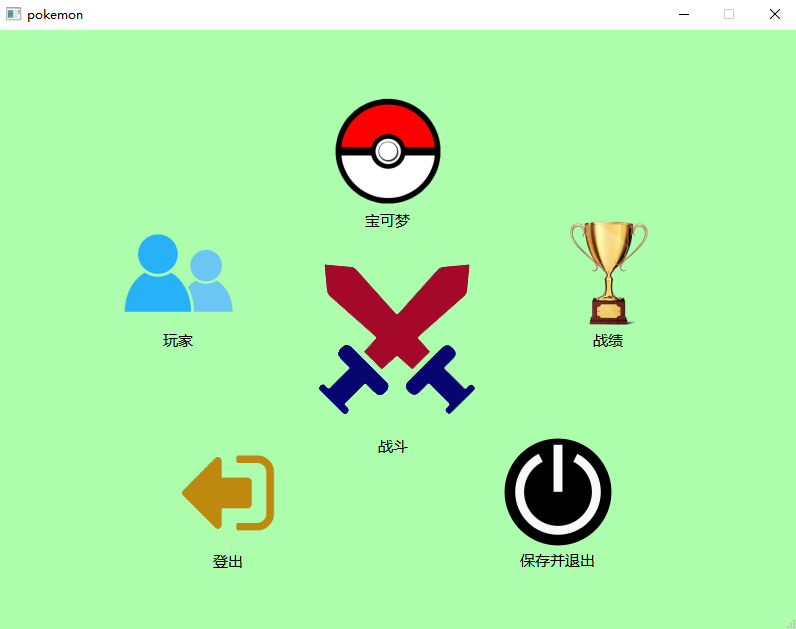
任意一方触发暴击或闪避时，界面同样会给予显示。

怒气值在每次受到攻击时即会增加1-2点，显示在宝可梦的下方。

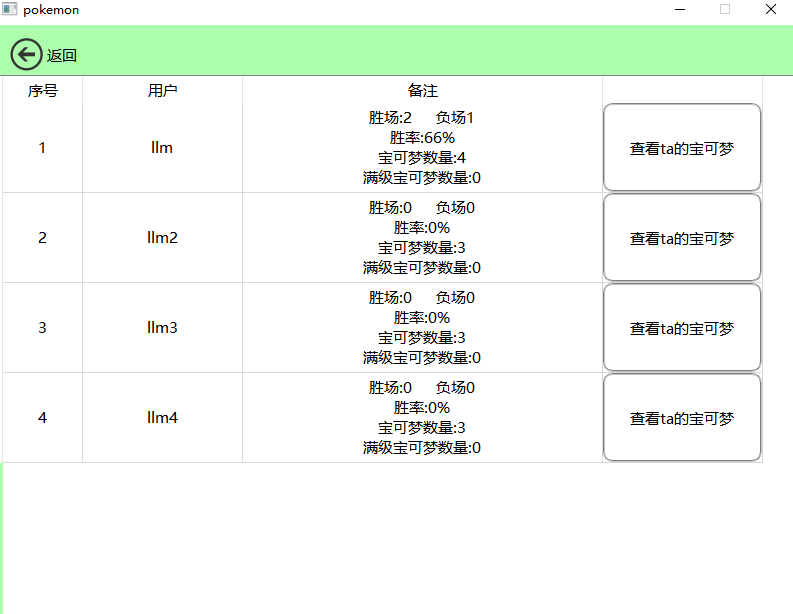
## 3、实现效果

注：部分功能已经在第二阶段实现，虽然在UI界面上有大幅优化，但是实际内容基本不变，因此这些界面下面仅仅列出，不予介绍。需要查看详细效果的话请参见目录“sourceCode\文档\stage3\_界面展示”下的图片。

### 3.1、前文已介绍的界面

登录界面 主界面

用户展示界面 宝可梦展示界面

### 3.2、成就界面



宝可梦徽章：拥有5只宝可梦时取得铜徽章，10只时取得银徽章，15只时取得金徽章。已获得徽章将会被点亮，未获得徽章显示为灰色。

高级宝可梦徽章：拥有5只满级（15级）宝可梦时取得铜徽章，10只时取得银徽章，15只时取得金徽章。已获得徽章将会被点亮，未获得徽章显示为灰色。

胜负场数、胜率：显示用户历史上参与的决斗的胜负情况。

返回：回到主界面。

### 3.3、战斗选择界面



三个界面依次为选择战斗类型、选择己方参战宝可梦，选择敌方参战宝可梦。

点击右上角问号图标可以查看战斗规则。点击返回图标回到上一界面。

### 3.4、战斗界面



对战双方：左方为己方，右方为敌方。鼠标移至双方图片上时可以看到各自的技能效果。

属性值：双方图片的上方显示各自的各项属性值。

怒气值（下方火苗图标）：每次受击获得1-2点怒气，怒气值满可释放技能。

普攻：我方回合时可以进行普通攻击。

技能：在我方回合时，怒气值积累到5点即可释放技能。鼠标移至技能图标上时可以看到己方的技能效果。

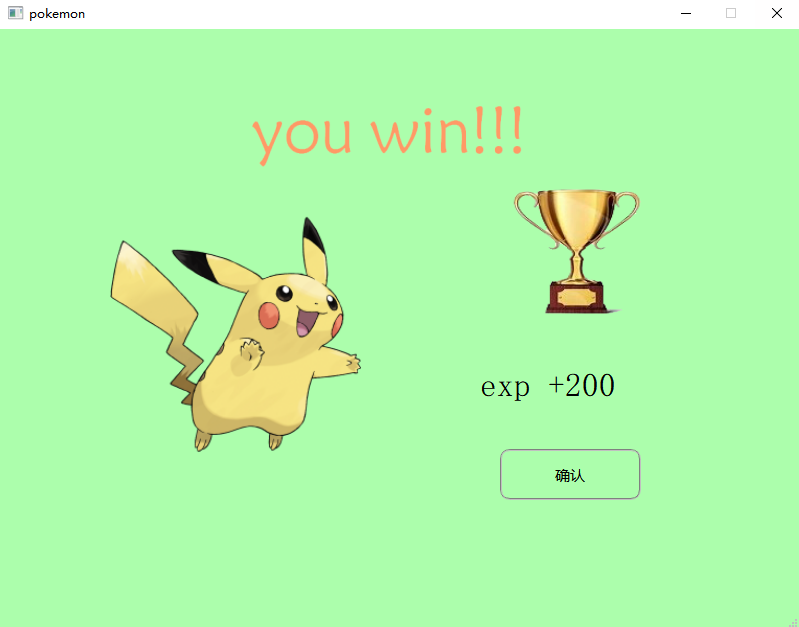
自动战斗：点击自动战斗图标启用自动战斗功能，再次点击后取消。

返回：退出战斗，回到主界面。点击退出图标后弹出提示框询问用户是否确认退出。

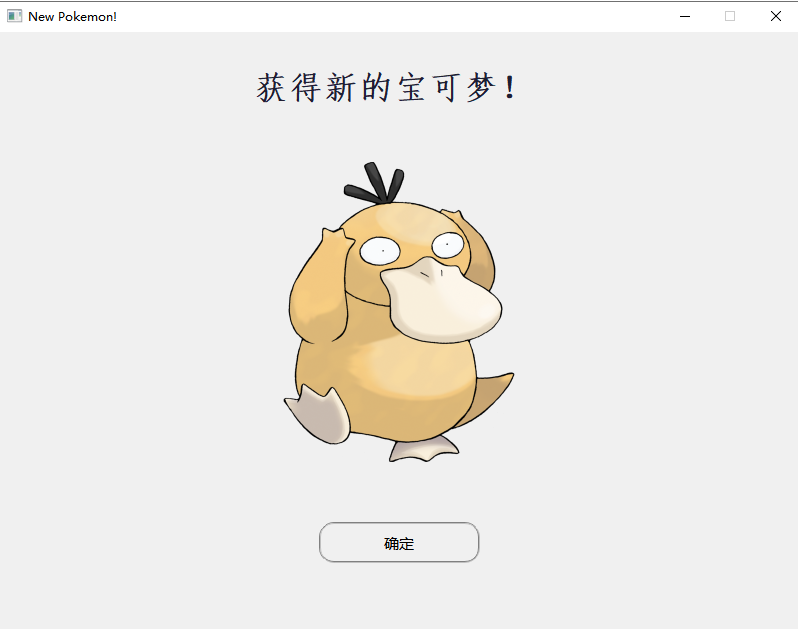
提示（右上角问号图标）：点击后查看战斗规则。

### 3.5、战斗结算界面

#### 3.5.1、胜利



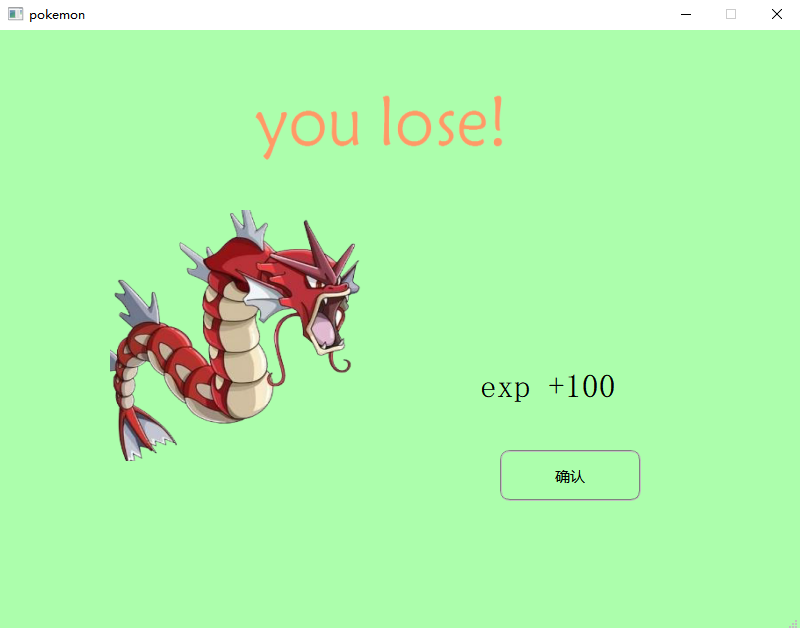
胜利界面



决斗赛中胜利可以

获得战胜的宝可梦

#### 3.5.2、失败

失败界面  
  


决斗赛中失败需要选

择一只宝可梦送出



失去所有宝可梦后

，系统补充一个

# 心得体会

经过一个多月陆陆续续的开发，今天终于完成了这个项目。虽说最终实现的画面依然很粗糙，功能也极其简单，但从基本为零的软件工程基础开始，到现在确确实实做出来一个东西，多少还是有点成就感的。

还记得一开始着手这个项目时，一大堆问题简直是让我焦头烂额。首先在图形界面开发工具Qt的使用上，我还相当不熟悉，基本上是想做一个功能（比如说定时器与多线程）但发现完全不会，只能做到哪百度到哪。由于Qt的功能相当强大，相应地各种控件使用起来也有极多细节，因此必须得花费相当多的时间进行研究。好不容易掌握了基本的语法后，又有两座大山出现在我面前：socket网络编程和数据库。这两项内容我之前没有接触过，刚开始做的时候碰到的一堆bug不知道让我崩溃了多少次，好在终究还是坚持了下来。随着开发进度的推进，我也不断变得更加熟练，懂得了怎么快速找到解决问题的方法。

这个游戏的设计应该算是我完成的第二个项目。相比第一次开发软件做旅行模拟系统（数据结构大作业）时，我确实感觉到自己是有一些进步的。第一次开发时对软件开发流程不懂，也没有深刻认识到面向对象思维在较大系统维护上的优越性，因此越到后期整个软件的体系架构就越显得杂乱无章，甚至到了加一个功能就要重写一大堆代码的地步。有了第一次的教训，再加上在C++课堂上学到的知识，在开发这个游戏时，我自然就更能把握整体设计与模块划分。尽管代码量从第一次的2000行变成了现在的5000行，但其实反而感觉比原来轻松一些了。

这次开发实践，既让我巩固了C++的基础知识，对类的继承、多态和封装有了更为深刻的认识，又让我接触到网络编程、数据库以及图形界面开发的世界，更是给了我一次动手实践的机会，培养了我解决问题的能力，为我今后进一步的学习和工作积累了经验。最重要的是，它让我体会到了编程的乐趣、制作游戏（虽然这游戏过于简单，不算严格意义上的游戏）的乐趣。今后，我定将更加努力，不断学习，把所学到的理论知识用于现实生活当中去。

# 附录1：战斗规则

1、战斗分为升级赛和决斗赛，两者都可以为宝可梦增加经验值

升级赛：

①选择宝可梦与服务器的宝可梦进行虚拟战斗。

②仅用于给宝可梦获得经验，胜利或失败不会得到或失去宝可梦。

决斗赛：

①选择宝可梦与服务器的宝可梦进行虚拟战斗。

②如果胜利，可以获得该战胜的宝可梦。

③如果失败，将由系统先随机选出3只宝可梦（如果用户拥有的宝可梦少于三只则选出其所有宝可梦），然后用户选择其中一只送出。

④当用户失去所拥有的最后一只宝可梦时，系统会自动发放一只初始状态的宝可梦。

2、伤害计算机制

面板攻击：宝可梦的基础攻击值。

基础倍率：0.90~1.10之间的一个随机值

技能倍率：普通攻击时为1，释放技能时请参见相关技能描述

暴击伤害倍率：触发暴击时为1.5，未触发为1

攻击造成的最终真实伤害=

面板攻击\*基础倍率\*技能倍率\*暴击伤害倍率-敌方防御值

3、暴击、闪避机制

暴击触发概率：20%

暴击伤害倍率：1.5

闪避触发概率：10%

闪避减免伤害：100%（即触发闪避时受到的伤害为0）

部分宝可梦的技能会改变触发暴击或闪避的概率，具体请参见技能描述

4、怒气值

每次受到攻击（包括触发闪避的情况）可以获得1~2点怒气值。

当怒气值达到5点时，可以使用技能。

释放技能后，怒气值清零。

5、技能（今后随版本更新可能会增加更多技能）

"rage"    愤怒：攻击力提高100%，且100%触发暴击(火系）

"tenacity"  顽强：每损失3%最大生命，下次攻击造成的伤害提高10%（土系）

"refresh"  回复：回复已损失生命的30%（水系）

"rebound"  反弹：对敌方造成上次被攻击所受伤害250%的伤害 (电系）

战斗时，鼠标移至技能图标或我方宝可梦图片上，可以显示我方宝可梦的技能。鼠标移至敌方宝可梦图片上，可以显示敌方宝可梦的技能。

6、死亡判定

我方宝可梦生命值降低到0时，战斗失败。

敌方宝可梦生命值降低到0时，战斗胜利。

7、自动战斗

点击自动战斗按钮进入自动战斗状态，再次点击后取消。

自动战斗状态下，不可以手动进行普通攻击或释放技能。

自动战斗状态下，敌我双方均为优先进行释放技能（即怒气值达到5点后立即释放技能）。

# 附录2：开发日志

开发过程分为3个阶段，对应于课程设计的三道题目：宠物小精灵的加入、用户注册与平台登陆、游戏对战的设计。

每个阶段的最终版本在下表中用红色加粗标识。最终发布版本为**V4.0**版本。

注：下文中，“宝可梦”与“宠物小精灵”、“精灵”为同一含义（宠物小精灵又名口袋妖怪，即pocket monster，缩写为pokemon，再音译为“宝可梦“）。

|  |  |  |
| --- | --- | --- |
| 版本 | 完成时间 | 完成内容 |
| **V1.0** | 7.2 | 1.完成宠物小精灵基类和四个精灵属性子类以及属性子类下共12种精灵种类子类的设计。  2.完成经验等级系统、攻击函数等设计。 |
| V2.0 | 7.26 | 1.采用了C/S模式，客户端与服务器通过UDP进行通信（采用了Qt中的QUdpSocket类）。  2.初步实现了注册、登录、登出功能。 |
| V2.1 | 7.27 | 1.使用了SQLite数据库保存用户的账户信息。 其中，客户端展示用户交互界面，服务器分别建立了账号数据库、宝可梦数据库，用于存储用户信息并响应客户端的请求。  2.完成了初次注册账号时系统自动分发宝可梦的功能。  3.完成了查看自己宝可梦详细信息的功能。  4.完善了注册、登录、登出等功能。 |
| **V2.2** | 7.28 | 1、实现了查看所有当前在线用户的功能。  2、实现了查看其它用户宝可梦信息的功能。  3、登出账号或退出游戏时，将会更新服务器端的数据库。  4、优化了UI设计。 |
| V3.0 | 8.13 | 1、大幅度优化了UI界面与图标。  2、实现了进入决斗界面以及决斗界面初始化的功能（包含选择我方出击的宝可梦和要进行对战的敌方宝可梦）。 |
| V3.1 | 8.14 | 1、初步设计了战斗的规则。  2、初步设计了行动条系统，决定每回合敌我双方攻击的顺序。  3、实现了战斗中的普通攻击功能（特殊技能有待后续开发）。 |
| V3.2 | 8.16 | 1、推翻了之前设计的行动条系统（由于实现动画效果过于复杂而不得已放弃），大幅改进了战斗的系统和规则。  2、增加了暴击和闪避机制。  3、增加了记录和查看玩家胜负场次数、胜率的功能。  4、增加了记录和查看玩家拥有的宠物数和高级宠物数的功能。  5、改进了客户端服务器数据交互的设计，改进了数据库的存储格式。  6、实现了决斗赛中：胜利可以获得该战胜的宝可梦，失败将失去一只宝可梦的功能。  7、实现了失去全部宝可梦后系统发放一只宝可梦的功能。 |
| V3.3 | 8.17 | 1、实现了一些简单的技能机制（目前技能数有限，后续视开发复杂程度可能进行改进）。  2、实现了一个简单的自动战斗系统。  3、实现了勋章系统。  4、优化了与战斗相关的设计。 |
| V3.4 | 8.18 | 1、增加了怒气值系统。怒气值与技能使用相关。  2、完善了自动战斗系统。 |
| **V3.5** | 8.19 | 1、修复了很多小bug。  2、进一步优化了相关界面设计。  3、补充了很多注释。  4、对代码进行进一步的规范。 |
| **V4.0** | 8.26 | 1、完善文档的编写。  2、修复bug。  3、确定最终版本。 |